Yeh, C. T., Brunette, T. J., Baker, D., McIntosh-Smith, S., & Parmeggiani, F. (2018). Elfin: an algorithm for the computational design of custom three-dimensional structures from modular repeat protein building blocks. *Journal of Structural Biology*, *201*(2), 100-107. https://doi.org/10.1016/j.jsb.2017.09.001

Title
Elfin: an algorithm for the computational design of custom three-dimensional structures from modular repeat protein building blocks

Authors
Chun-Ting Yeh [1]
TJ Brunette [2]
David Baker [2]
Simon McIntosh-Smith [1]
Fabio Parmeggiani [3,4]

[1] Department of Computer Science, University of Bristol, UK
[2] Department of Biochemistry, Institute for Protein Design, University of Washington, Seattle, WA, USA
[3] School of Chemistry, School of Biochemistry, University of Bristol, UK
[4] BrisSynBio, BBSRC/EPSRC Synthetic Biology Research Centre, Bristol, UK

Corresponding author: Fabio Parmeggiani, fabio.parmeggiani@bristol.ac.uk

Keywords
Repeat protein
Protein design
Protein origami
Genetic algorithm
Computational protein design

Abstract
Computational protein design methods have enabled the design of novel protein structures, but they are often still limited to small proteins and symmetric systems. To expand the size of designable proteins while controlling the overall structure, we developed Elfin, a genetic algorithm for the design of novel proteins with custom shapes using structural building blocks derived from experimentally verified repeat proteins. By combining building blocks with compatible interfaces, it is possible to rapidly build non-symmetric large structures (> 1000 amino acids) that match three-dimensional geometric descriptions provided by the user. A run time of about 20 minutes on a laptop computer for a 3000 amino acid structure makes Elfin accessible to users with limited computational resources. Protein structures with controlled geometry will allow the systematic study of the effect of spatial arrangement of enzymes and signaling molecules, and provide new scaffolds for functional nanomaterials.

1. Introduction

The precise design of nanometer-size protein structures is becoming increasingly important with the emphasis on miniaturized devices and interest in spatial organization of enzymes and molecular binders (Glover and Clark, 2016), but so far custom structures have been mainly designed with DNA nanotechnology (Zhang et al., 2014).
The complexity of interactions within proteins limits the ability to reliably generate new structures with custom shapes (protein origami), but computational methods can now

accurately design *de novo* proteins without relying on template structures (Gradišar et al., 2013; Huang et al., 2014, 2016a; Marcos et al., 2017), or combine multiple polypeptide chains into oligomers (Thomson et al., 2014), symmetric cage-like structures (King et al., 2012, 2014; Lai et al., 2014) and lattices (Gonen et al., 2015; Lanci et al., 2012).
De novo proteins are designed in two steps: backbones are sampled through peptide fragments (Koga et al., 2012; Kuhlman et al., 2003; Lin et al., 2015), kinematic closure (Bhardwaj et al., 2016; Coutsias et al., 2004), combination of secondary structure elements (Jacobs et al., 2016), and parametric equations (Crick, 1953; Grigoryan and DeGrado, 2011), followed by diverse search strategies for backbone compatible sequences, recently reviewed (Gainza et al., 2016).
Sequence length heavily influences design complexity, increasing the number of pairwise interactions to be sampled and the time required. Larger designs were achieved by using repeated structural motifs within the polypeptide (Brunette et al., 2015; Doyle et al., 2015; Huang et al., 2016b; Parmeggiani et al., 2015).
In these cases, complexity is not simply reduced by constraining structurally equivalent positions to the same amino acid. Contacts between residues along the chain are limited, since generally each repeat interacts only locally with its two neighboring units, located before and after it in the primary sequence (Javadi and Itzhaki, 2013; Kajava, 2012; Paladin et al., 2017; Parmeggiani and Huang, 2017). Combinations of different repeats with compatible interfaces allow control of the overall protein shape (Park et al., 2015). These interfaces ensure that a continuous hydrophobic core is formed across the structure and that adjacent units are precisely oriented.
We exploited repeat protein features to develop Elfin, a genetic algorithm (GA) for assembly of compatible repeat protein building blocks. By using structurally characterized units with known sequences, we drastically reduce the search space and enable the design of large structures with custom shapes using geometric specifications, similarly to what occurs for DNA nanostructures (Douglas et al., 2009; Veneziano et al., 2016).

2. Material and methods

Programming languages and library dependencies

- **Python** for auxiliary tasks such as database pre-processing, benchmark generation, and output post-processing. Libraries: biopython, pymol, numpy, matplotlib.
- **C++(11)** for the main GA. Libraries: STL, JSON by nlohmann (open source), and in-house C utility library (open source)

Elfin was ported to OpenMP4.0 to take advantage of multithreaded processing on both CPUs and GPUs.
The code and the JSON database containing centers of mass and geometric relationships are publicly available at https://github.com/joy13975/elfin.
Genetic algorithm solutions can be converted into center of mass representations. The structural database will be released upon publication of the designed structures it contains, allowing users to convert solutions to full-atom pdbs.

2.1 Database generation

2.1.1 Module pre-processing

The generation of compatible modules requires uniform interfaces. We used the repacked and minimized interface between internal repeats of designed helical repeat proteins

(Brunette et al., 2015) as our reference, and enforce backbone and side chain dihedral angles on all equivalent interfaces.
Modules were repacked and minimized with the fast relax protocol of the Rosetta modelling suite (Leaver-Fay et al., 2011; Tyka et al., 2011) while holding the interfaces fixed. Modules with a root mean square deviation of more than 1 Å after relax were not included in the database, to avoid potential flexibility issues. Through this process, the N and C termini of compatible modules are already located in the correct position to form a peptide bond and do not require additional design.

2.1.2 Database abstraction

For each single module PDBs, we first compute the center of mass (CoM) using carbon-alpha (Cα) coordinates, and then translate the module to place the CoM at the origin of the Cartesian coordinate system. The alignment to the origin reduces the number of operations at the building stage, increasing the speed. The module size is indicated by the radius of gyration, defined as the average Cα distance from the CoM. Each module pair is aligned to the centered version of its first module using Kabsch's absolute orientation algorithm (Kabsch, 1976). For each pair, we then compute the CoM of each single module, and the translation T and rotation R for aligning the pair to the centered version of its second module. The CoMs, R, and T are stored into a new "abstraction" database. Hereafter, all mentions of "pairs" and "singles" are respectively the aligned and the centered versions (pre-processed) of their original PDB representations. These pre-processed versions are just 3D points (CoMs), related by their corresponding R and T. The "abstraction" database is stored in JSON format and also contains the information about the compatible modules.

2.2 Target representation and protein building from abstraction database

3D trajectories describing target shapes were specified using a Matlab script, available in the github repository, which allows the user to plot points and scale the shape. However, any other tool that generates a sequence of points in 3D space will be suitable.
Module sequences for the genetic algorithm's initial population and benchmark shapes were built by sequential addition of randomly selected compatible modules. From the sequence of modules, we can build its 3D point-representation using the abstraction database. The shape is built by sequential placement of CoM pairs, using the second CoM of the last added pair to place by superposition the first CoM of the next pair ("place" step). The chain is then re-oriented with the new CoM at the axis origin ("push" step). Each step of chain growth is a matrix transformation $X' = RX + T$, where X is the array of 3D CoMs currently representing the shape we are constructing, followed by the addition of a new CoM (the second of the last added pair). The geometry of the overall shape is described by the sequence of CoMs for the constituent modules. At each step, collision checks are performed to ensure that the radii of gyration are not violated when placing the next module.

2.3 Genetic Algorithm features

Elfin is a Genetic Algorithm (GA) that iterates through mutation, scoring, ranking, and selection. Each individual is a vector of protein module identifiers. The initial individuals and any new sequence introduced at each cycle are generated as described in section 2.2.
In addition to the population size and maximum iteration settings, Elfin contains the following tunable parameters that affect the quality of the solution and the execution time:

• Average Pair CoM Distance: the average CoMs distance of all database pairs is used to compute an expected sequence length for real design inputs. The value is derived from the input abstraction database, 45Å in our case.
• Length Deviation allowance: the number of modules from the expected sequence length that an individual is allowed to deviate.
• Survival Rate: the ratio of population to survive each generation.
• Crossover Rate: the ratio of non-survivors to undergo crossover.
• Point Mutate Rate: the ratio of non-survivors to undergo point mutation.
• Limb Mutate Rate: the ratio of non-survivors to undergo limb mutation.
• Score Stop Threshold: the score below which the GA should consider the solution acceptable and stop. We kept it at a value of 0 (perfect matching) for all our cases.
• Maximum Stagnant Generations: the number of iterations without score improvement after which the GA exits.

2.4 GA scoring

In our context, scoring is a shape analysis problem that requires matching of rotation and translation, but not scale. Each individual design is to be scored against the input shape specification, which means assessing a generally imperfect superimposition of the 3D CoMs. This is in fact the partial Procrustes problem (Schönemann, 1966), usually solved by Kabsch's absolute orientation algorithm (Kabsch, 1976). However, Kabsch's algorithm only works for two shapes that have an equal number of points. Since we can only estimate the design length for any given shape specification, and that each individual design in GA is allowed a deviation from that expectation, a method to compare differently-sized 3D CoMs arrays is needed. We developed a proportional re-sampling subroutine to equalize the number of points defining the two shapes in question. Elfin up- or down- samples the designed shape to the number of points present in the target. Distances between points are proportional to the distances in the input shape. A perfect superposition has a score of 0, with larger positive values indicating increasing deviations.
Finally, ranking is simply sorting the individuals in ascending order of score. Care was taken to enforce diversity, meaning no repeating parents should survive in the same generation. This was achieved using a cyclic redundancy check (CRC), conveniently computed during the generation of each individual design.

2.5 Benchmarks and optimization:

We randomly generated from the database (see section 2.2) shape specifications of various lengths (10 made of 10 modules, 5 of 20 modules and 5 of 30 modules) to test the effectiveness of Elfin.
Two grid searches were conducted in order to determine optimal parameters for the genetic algorithm. In the first search, a wide spread of discrete values were tested on benchmarks and on real designs. The Pareto Front taking target similarity score and execution time as objectives was identified to guide a second search that covered a narrower spread of discrete values.
After two searches the configuration did not seem to shift from the first Pareto Front. The parameters we found to reach exact benchmark solutions in the shortest time are below:
Length deviation allowance: 0.2 (20%)
Population survival rate: 0.02 (2%)
Crossover rate (of non-survivors): 0.2 (20%)

Point mutate rate (of non-survivors not crossed over): 0.4 (40%)
Limb mutate rate (of non-survivors not crossed over): 0.4 (40%)
The numbers in parenthesis indicate the percentage of the population at each iteration. The remaining fraction is filled with new randomly generated individuals.

2.6 Performance evaluation

A runtime measurement of Elfin was conducted across several CPUs and GPUs. Since Elfin makes heavy use of random number generators and single precision floating point operations, executing on different platforms or being compiled by different compilers could introduce a difference in the total number of iterations required for solving a particular design problem. Therefore, during our experiments Elfin was configured to always process 50 generations, with a population of 524288, and use the 'B' letter shape input. This number of generations ensured the best solution was found for this particular design on each platform, while keeping the amount of computations roughly equal. On each platform, we compiled Elfin with GCC 6.1.0 without architecture-specific optimization flags.

3. Results

We developed a pipeline to describe and manipulate structures in an abstract representation, allowing us to rapidly compare designs to target shapes. The algorithm builds a single polypeptide by linking structural building blocks.
The pipeline stages are (Fig.1):

1. Collection of structurally characterized repeat proteins (section 3.1)
2. Assembly of a database of compatible structural modules (section 3.2)
3. Description of modules and target shapes in a simplified abstract representation (section 3.3)
4. Search for module combinations that satisfy the target shape (section 3.4)
5. Transformation of abstract representations of solutions in structural models and validation (section 3.5)

Upon optimization of the performance on a benchmark test set, we used Elfin to design custom three-dimensional structures (section 3.6).

3.1 Experimental data

The design process relies on building blocks that are directional, interacting specifically with the previous and the following modules in a structure, as repeats in solenoid-like proteins (Kobe and Kajava, 2000). Among repeat proteins, designed helical repeat proteins (DHRs) (Brunette et al., 2015) are highly stable and display a broad range of structural diversity that can be incorporated in our modular system. For each protein, internal repeats are identical. Small angle x-ray scattering (SAXS) data indicate that designed repeat proteins maintain in solution the same conformation observed in crystal structures and the overall shape can be correctly predicted even upon an increase in the number of repeats (Fallas et al., 2017), therefore they can be considered rigid on a first approximation. Since the interface between repeats is responsible for compatibility, it is possible to generate different structures that display the same interface. In analogy to previous work on leucine rich repeats (Park et al., 2015), proteins containing two different types of DHR repeats at the N and C termini were designed, expressed and characterized by SAXS (manuscript in preparation). A non-repeated portion bridges the two repeat interfaces and maintains a continuous hydrophobic core.

These designs are referred to as junctions and provide an additional source of structurally validated proteins for our database.

3.2 Module database

The building blocks we used in our database are called modules. Base modules correspond to the central portion of characterized proteins, after removing the N- and C-terminal repeats that shield the hydrophobic core from the solvent (Fig.2A and 2B). For DHRs, each module is constituted by the two identical central repeats. DHR repeats are characterized by helix-loop-helix-loop structural motifs. Junction modules are derived from experimentally validated fusions between different repeat units with shared hydrophobic core and secondary structure elements (manuscript in preparation) (Fig.2C). Junction modules allow us to connect different types of DHRs (Fig.2D). The database includes 35 single modules (singles), 12 from DHRs and 23 from junctions, and 143 module pairs (pairs) that describe interactions between two modules, either DHR-DHR, DHR-junction or junction-junction. Modules to include in the database were selected as described in section 2.1.1.

3.3 Abstraction and design specification

Modules are considered rigid building blocks and represented by a center of mass (CoM) and a radius. Interacting pairs, based on the presence of compatible interfaces, are described by the translation and rotation of the second CoM in relation to the first (Fig.1 and section 2.1.2). The final design is then represented as a series of CoM in 3D space related by rigid body transforms. Similarly, a target shape is described by a sequence of points in space and their order of connectivity.

3.4 Genetic algorithm (GA)

The Elfin GA iterates between mutation, scoring, ranking, and selection. In contrast to previous GA applications in protein design (Jones, 1994; Lazar et al., 1997; Pedersen and Moult, 1996; Unger and Moult, 1993), we do not search for amino acid sequences or minimize the energy of the system, but explore sequences of modules that most closely describe the target shape. However, the number of combinations increases rapidly with the size of the target, the modules in the database and the number of pairs available. The complexity for a 30 modules design with our current database is approximately 10^{15}.
The initial population for the GA is randomly assembled from pairs in the abstraction database, which contains only allowed module combinations (see section 2.2). The target length is calculated as the sum of distances between consecutive points. A length deviation allowance is included to ensure a broad range of starting designs with different number of modules. The designs are built sequentially by aligning the first module of a pair to the last module of the growing chain and reorienting the chain in space. The series of “place” and “push” steps are shown in Fig.3a and described in section 2.2. Designs are scored by similarity to the target (see sections 2.3 and 2.4). At each GA cycle, the high scoring individuals are promoted to the next iteration, while others are recombined by crossing over or mutated by replacing modules (Fig.3b) and new combinations are assembled and added to the pool.
The mutation stage employs three operators:

1. Crossover: two randomly selected parents are checked for compatible crossing modules. Once found, a child sequence is produced by combining complementing “limbs” of its parents.

2. Point-mutate: performs an insertion, deletion, or alteration of one module in the sequence.
3. Limb-mutate: from a randomly chosen point in the module sequence, a randomly chosen side is erased and that "limb" is then re-grown randomly.

The GA exits with a solution when it reaches the maximum allowed number of iterations, the score falls below a set threshold or the lowest score does not change anymore (stagnation). The GA default output contains the three lowest scoring solutions but the number can be increased.

3.5 Conversion to protein models and minimization

In our tests, repeated runs converged to the same lowest scoring solution, which we converted into a full atom model using the modules sequence and the CoMs coordinates. For complex designs where no convergence is observed, multiple solutions can be analyzed. The process is similar to GA shape construction, except now PDBs from the module database are used. Structures are then repacked and minimized iteratively using the fast relax protocol in Rosetta (Tyka et al., 2011). Due to potential lever arm effects – small deviations in modules can result in large overall deviation from the pre-relax structure because of the length of the chain after that point – we evaluated the fidelity of the relaxed design using windows of 300 residues with overlaps of 150 amino acids. If all windows have RMSD below 5 Å we consider a design successful.

3.6 Designing structures

We first assessed the ability of Elfin to find solutions for target shapes generated randomly from our abstraction database: 10 with 10 modules (RL10), 5 with 20 modules (RL20) and 5 with 30 modules (RL30). In all 20 cases, Elfin converged to the exact solution in a single run within 100 GA generations. We then assessed Elfin's capabilities of designing proteins according to 10 hand-drawn (HD) target shapes. Fig.4 illustrates random test shapes and hand drawn shapes results, for the best and worst targets in terms of RMSD before and after relaxation. All the designs converged to a single solution within 1 run of Elfin, achieving striking visual similarity to the target specification.

We then used the benchmark set and the hand drawn targets for optimization of the GA parameters (see section 2.5). We evaluated Elfin's performance across hardware platforms, from consumer grade laptops to server CPUs to GPUs. Fig.5 shows the run time of a "B" shape with fixed population and number of generations (see section 2.6).

4. Discussion

Elfin aims to control the overall shape and size of a protein using pre-existing building blocks, in contrast to other approaches that aim to specifically design every residue. By using known compatible interfaces between modules, no sequence design step takes places, drastically reducing computing time. This approach allowed us, for the first time, to build models for single chain proteins up to 5000 residues in size.

Elfin was able to correctly solve all 20 randomly generated tests (i.e. to find the exact sequence of modules) after just one run per input. During these runs, the GA was set to stop if a score of zero (perfect superposition with the target) was found, or if 50 generations went by without any score improvement. For the RL30 sequences, Elfin searched no more than two hundred generations with 524,288 sequences in the population. Out of an

estimated 10^{15} possible combinations in RL30 sequences only a total of less than 10^{9} had to be sampled before Elfin converged. This is evidence of the successful implementation of the GA and the effective design of its mutagenesis operators.
Elfin's speed allows its use on personal computers, where a more than 3000 residue protein can be designed on a 2.6 GHz 4 core processor in less than 20 minutes (Fig.5). The algorithm scales for clusters and GPUs, with GTX1080 outperforming all of the CPUs tested. This was not surprising as Elfin is a compute intensive, easily parallelized code. However, we are aware of limitations in the current GPU implementation of Elfin. The first of these is workload imbalance, which leads to less efficient work sharing and overheads in scheduling. Another known issue in the GPU version of Elfin is that it repeatedly copies the entire population to and from the target GPU device memory. GPU optimization will address these issues and further reduce the time required to reach a solution.
The major assumption in our approach is the rigidity of the building blocks. Although building blocks were selected from experimentally characterized structures, their flexibility in the context of larger proteins was not experimentally verified. The Rosetta relax protocol highlighted that building block deformations can occur within designed structures. However, even in the worse performing cases, according to our RMSD cut-off, the shape is still clearly recognizable. This is possible because errors are confined: generally large RMSD deviations are related to distortion in single modules after relax, indicating a problem in local stability. A more extensive search, beyond the scope of the current work, will allows us to identify these modules and establish whether the distortion is influenced by specific neighboring modules. Moreover, experimental verification of designed structures will provide valuable information about shape and rigidity that could feed back into the database (e.g. modules or pairs associated with distortion or solubility issues).
The designs are currently limited to single protein chains with "open" structures, but we are planning to extend elfin for building "close" structures, where N- and C-terminal modules interact. The database can be expanded with new structural (e.g. oligomers) and functional modules (e.g. enzymes or domains binding proteins, nucleic acids or small molecules), allowing the design of novel protein architectures.

5. Conclusions

Elfin was developed to design large protein structures with custom shapes. Relying on characterized building blocks reduces the search space, eliminates sequence design steps and produces three-dimensional structures corresponding to user descriptions. The speed and parallelization allow users to rapidly build multiple designs to be assessed experimentally. This approach represents a new avenue for the generation of novel proteins with specific shapes and could be used to investigate the effect of spatial organization of enzymes, DNA and signaling molecules *in vitro* and *in vivo*. Moreover, the abstract nature of the design process, requiring only geometric descriptions, opens up the possibility to build custom structures not only from repeat proteins, but also from other modular system.

Acknowledgements

We would like to thank the Advanced Computing Research Centre (ACRC) and BrisSynBio, a BBSRC/EPSRC Synthetic Biology Research Centre, at the University of Bristol for access to the BlueCrystal and Bluegem supercomputers. We would also like to thank Intel and the Intel Parallel Computing Center at the University of Bristol for access to the KNL system we

used for testing. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Bhardwaj, G., Mulligan, V.K., Bahl, C.D., Gilmore, J.M., Harvey, P.J., Cheneval, O., Buchko, G.W., Pulavarti, S.V.S.R.K., Kaas, Q., Eletsky, A., Huang, P.-S., Johnsen, W.A., Greisen, P.J., Rocklin, G.J., Song, Y., Linsky, T.W., Watkins, A., Rettie, S.A., Xu, X., Carter, L.P., Bonneau, R., Olson, J.M., Coutsias, E., Correnti, C.E., Szyperski, T., Craik, D.J., Baker, D., 2016. Accurate de novo design of hyperstable constrained peptides. Nature 538, 329–335. doi:10.1038/nature19791

Brunette, T.J., Parmeggiani, F., Huang, P.-S., Bhabha, G., Ekiert, D.C., Tsutakawa, S.E., Hura, G.L., Tainer, J.A., Baker, D., 2015. Exploring the repeat protein universe through computational protein design. Nature 528, 580–584. doi:10.1038/nature16162

Coutsias, E.A., Seok, C., Jacobson, M.P., Dill, K.A., 2004. A kinematic view of loop closure. J. Comput. Chem. 25, 510–528. doi:10.1002/jcc.10416

Crick, F.H.C., 1953. The Fourier transform of a coiled-coil. Acta Crystallogr. 6, 685–689. doi:10.1107/S0365110X53001952

Douglas, S.M., Marblestone, A.H., Teerapittayanon, S., Vazquez, A., Church, G.M., Shih, W.M., 2009. Rapid prototyping of 3D DNA-origami shapes with caDNAno. Nucleic Acids Res. 37, 5001–5006. doi:10.1093/nar/gkp436

Doyle, L., Hallinan, J., Bolduc, J., Parmeggiani, F., Baker, D., Stoddard, B.L., Bradley, P., 2015. Rational design of α-helical tandem repeat proteins with closed architectures. Nature 528, 585–588. doi:10.1038/nature16191

Fallas, J.A., Ueda, G., Sheffler, W., Nguyen, V., McNamara, D.E., Sankaran, B., Pereira, J.H., Parmeggiani, F., Brunette, T.J., Cascio, D., Yeates, T.R., Zwart, P., Baker, D., 2017. Computational design of self-assembling cyclic protein homo-oligomers. Nat. Chem. 9, 353–360. doi:10.1038/nchem.2673

Gainza, P., Nisonoff, H.M., Donald, B.R., 2016. Algorithms for protein design. Curr. Opin. Struct. Biol., Engineering and design • Membranes 39, 16–26. doi:10.1016/j.sbi.2016.03.006

Glover, D.J., Clark, D.S., 2016. Protein Calligraphy: A New Concept Begins To Take Shape. ACS Cent. Sci. 2, 438–444. doi:10.1021/acscentsci.6b00067

Gonen, S., DiMaio, F., Gonen, T., Baker, D., 2015. Design of ordered two-dimensional arrays mediated by noncovalent protein-protein interfaces. Science 348, 1365–1368. doi:10.1126/science.aaa9897

Gradišar, H., Božič, S., Doles, T., Vengust, D., Hafner-Bratkovič, I., Mertelj, A., Webb, B., Šali, A., Klavžar, S., Jerala, R., 2013. Design of a single-chain polypeptide tetrahedron assembled from coiled-coil segments. Nat. Chem. Biol. 9, 362–366. doi:10.1038/nchembio.1248

Grigoryan, G., DeGrado, W.F., 2011. Probing Designability via a Generalized Model of Helical Bundle Geometry. J. Mol. Biol. 405, 1079–1100. doi:10.1016/j.jmb.2010.08.058

Huang, P.-S., Boyken, S.E., Baker, D., 2016a. The coming of age of de novo protein design. Nature 537, 320–327. doi:10.1038/nature19946

Huang, P.-S., Feldmeier, K., Parmeggiani, F., Fernandez Velasco, D.A., Höcker, B., Baker, D., 2016b. De novo design of a four-fold symmetric TIM-barrel protein with atomic-level accuracy. Nat. Chem. Biol. 12, 29–34. doi:10.1038/nchembio.1966

Huang, P.-S., Oberdorfer, G., Xu, C., Pei, X.Y., Nannenga, B.L., Rogers, J.M., DiMaio, F., Gonen, T., Luisi, B., Baker, D., 2014. High thermodynamic stability of parametrically designed helical bundles. Science 346, 481–485. doi:10.1126/science.1257481

Jacobs, T.M., Williams, B., Williams, T., Xu, X., Eletsky, A., Federizon, J.F., Szyperski, T., Kuhlman, B., 2016. Design of structurally distinct proteins using strategies inspired by evolution. Science 352, 687–690. doi:10.1126/science.aad8036

Javadi, Y., Itzhaki, L.S., 2013. Tandem-repeat proteins: regularity plus modularity equals design-ability. Curr. Opin. Struct. Biol. 23, 622–631. doi:10.1016/j.sbi.2013.06.011

Jones, D.T., 1994. De novo protein design using pairwise potentials and a genetic algorithm. Protein Sci. 3, 567–574. doi:10.1002/pro.5560030405

Kabsch, W., 1976. A solution for the best rotation to relate two sets of vectors. Acta Crystallogr. A 32, 922–923. doi:10.1107/S0567739476001873

Kajava, A.V., 2012. Tandem repeats in proteins: From sequence to structure. J. Struct. Biol., Structural Bioinformatics 179, 279–288. doi:10.1016/j.jsb.2011.08.009

King, N.P., Bale, J.B., Sheffler, W., McNamara, D.E., Gonen, S., Gonen, T., Yeates, T.O., Baker, D., 2014. Accurate design of co-assembling multi-component protein nanomaterials. Nature 510, 103–108. doi:10.1038/nature13404

King, N.P., Sheffler, W., Sawaya, M.R., Vollmar, B.S., Sumida, J.P., André, I., Gonen, T., Yeates, T.O., Baker, D., 2012. Computational Design of Self-Assembling Protein Nanomaterials with Atomic Level Accuracy. Science 336, 1171–1174. doi:10.1126/science.1219364

Kobe, B., Kajava, A.V., 2000. When protein folding is simplified to protein coiling: the continuum of solenoid protein structures. Trends Biochem. Sci. 25, 509–515. doi:10.1016/S0968-0004(00)01667-4

Koga, N., Tatsumi-Koga, R., Liu, G., Xiao, R., Acton, T.B., Montelione, G.T., Baker, D., 2012. Principles for designing ideal protein structures. Nature 491, 222–227. doi:10.1038/nature11600

Kuhlman, B., Dantas, G., Ireton, G.C., Varani, G., Stoddard, B.L., Baker, D., 2003. Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. Science 302, 1364–1368. doi:10.1126/science.1089427

Lai, Y.-T., Reading, E., Hura, G.L., Tsai, K.-L., Laganowsky, A., Asturias, F.J., Tainer, J.A., Robinson, C.V., Yeates, T.O., 2014. Structure of a designed protein cage that self-assembles into a highly porous cube. Nat. Chem. 6, 1065–1071. doi:10.1038/nchem.2107

Lanci, C.J., MacDermaid, C.M., Kang, S., Acharya, R., North, B., Yang, X., Qiu, X.J., DeGrado, W.F., Saven, J.G., 2012. Computational design of a protein crystal. Proc. Natl. Acad. Sci. 109, 7304–7309. doi:10.1073/pnas.1112595109

Lazar, G.A., Desjarlais, J.R., Handel, T.M., 1997. De novo design of the hydrophobic core of ubiquitin. Protein Sci. Publ. Protein Soc. 6, 1167–1178.

Leaver-Fay, A., Tyka, M., Lewis, S.M., Lange, O.F., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P.D., Smith, C.A., Sheffler, W., Davis, I.W., Cooper, S., Treuille, A., Mandell, D.J., Richter, F., Ban, Y.-E.A., Fleishman, S.J., Corn, J.E., Kim, D.E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J.J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J.J., Kuhlman, B., Baker, D., Bradley, P., 2011. ROSETTA3: an

object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol. 487, 545–574. doi:10.1016/B978-0-12-381270-4.00019-6

Lin, Y.-R., Koga, N., Tatsumi-Koga, R., Liu, G., Clouser, A.F., Montelione, G.T., Baker, D., 2015. Control over overall shape and size in de novo designed proteins. Proc. Natl. Acad. Sci. 112, E5478–E5485. doi:10.1073/pnas.1509508112

Marcos, E., Basanta, B., Chidyausiku, T.M., Tang, Y., Oberdorfer, G., Liu, G., Swapna, G.V.T., Guan, R., Silva, D.-A., Dou, J., Pereira, J.H., Xiao, R., Sankaran, B., Zwart, P.H., Montelione, G.T., Baker, D., 2017. Principles for designing proteins with cavities formed by curved β sheets. Science 355, 201–206. doi:10.1126/science.aah7389

Paladin, L., Hirsh, L., Piovesan, D., Andrade-Navarro, M.A., Kajava, A.V., Tosatto, S.C.E., 2017. RepeatsDB 2.0: improved annotation, classification, search and visualization of repeat protein structures. Nucleic Acids Res. 45, D308–D312. doi:10.1093/nar/gkw1136

Park, K., Shen, B.W., Parmeggiani, F., Huang, P.-S., Stoddard, B.L., Baker, D., 2015. Control of repeat-protein curvature by computational protein design. Nat. Struct. Mol. Biol. 22, 167–174. doi:10.1038/nsmb.2938

Parmeggiani, F., Huang, P.-S., 2017. Designing repeat proteins: a modular approach to protein design. Curr. Opin. Struct. Biol., Engineering and design • Membranes 45, 116–123. doi:10.1016/j.sbi.2017.02.001

Parmeggiani, F., Huang, P.-S., Vorobiev, S., Xiao, R., Park, K., Caprari, S., Su, M., Seetharaman, J., Mao, L., Janjua, H., Montelione, G.T., Hunt, J., Baker, D., 2015. A General Computational Approach for Repeat Protein Design. J. Mol. Biol. 427, 563–575. doi:10.1016/j.jmb.2014.11.005

Pedersen, J.T., Moult, J., 1996. Genetic algorithms for protein structure prediction. Curr. Opin. Struct. Biol. 6, 227–231. doi:10.1016/S0959-440X(96)80079-0

Schönemann, P.H., 1966. A generalized solution of the orthogonal procrustes problem. Psychometrika 31, 1–10. doi:10.1007/BF02289451

Thomson, A.R., Wood, C.W., Burton, A.J., Bartlett, G.J., Sessions, R.B., Brady, R.L., Woolfson, D.N., 2014. Computational design of water-soluble α-helical barrels. Science 346, 485–488. doi:10.1126/science.1257452

Tyka, M.D., Keedy, D.A., André, I., DiMaio, F., Song, Y., Richardson, D.C., Richardson, J.S., Baker, D., 2011. Alternate States of Proteins Revealed by Detailed Energy Landscape Mapping. J. Mol. Biol. 405, 607–618. doi:10.1016/j.jmb.2010.11.008

Unger, R., Moult, J., 1993. Genetic Algorithms for Protein Folding Simulations. J. Mol. Biol. 231, 75–81. doi:10.1006/jmbi.1993.1258

Veneziano, R., Ratanalert, S., Zhang, K., Zhang, F., Yan, H., Chiu, W., Bathe, M., 2016. Designer nanoscale DNA assemblies programmed from the top down. Science 352, 1534–1534. doi:10.1126/science.aaf4388

Zhang, F., Nangreave, J., Liu, Y., Yan, H., 2014. Structural DNA Nanotechnology: State of the Art and Future Perspective. J. Am. Chem. Soc. 136, 11198–11211. doi:10.1021/ja505101a

Figures

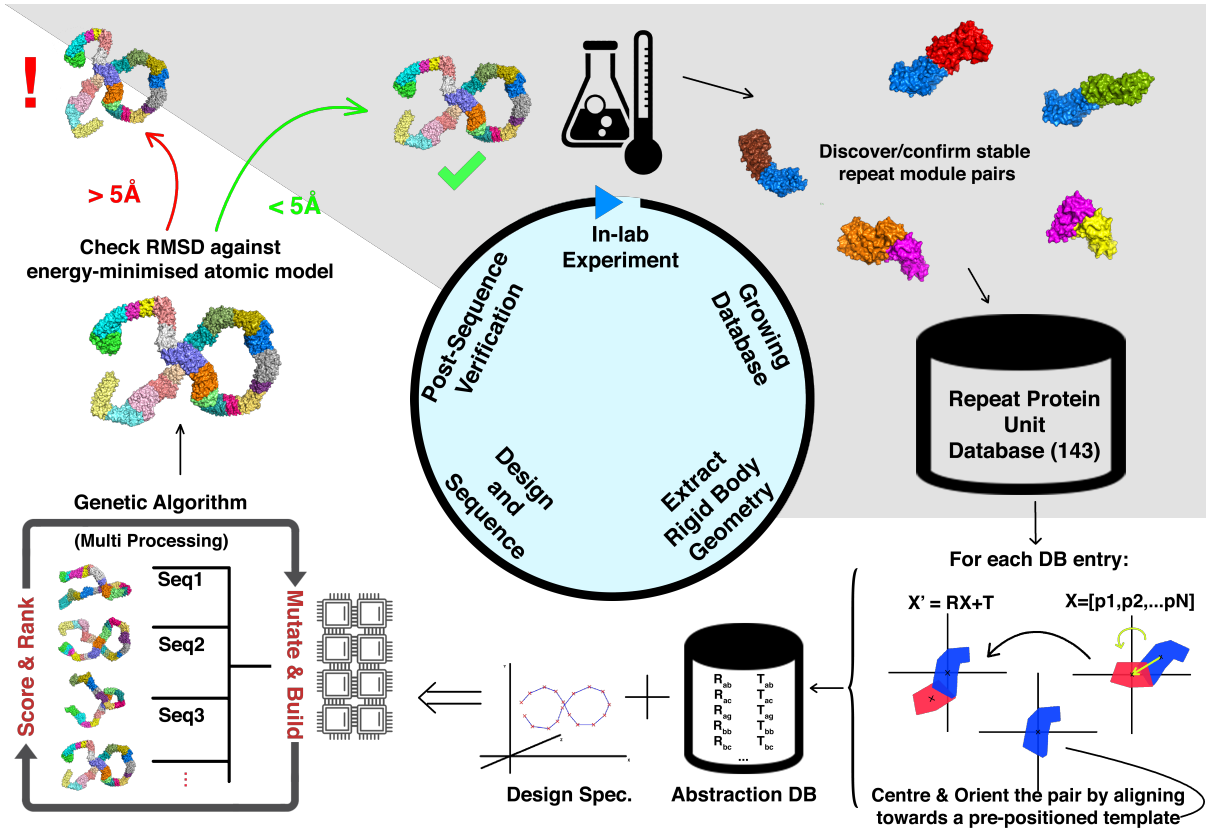


Figure 1

The modular design pipeline relies on structurally validated designed repeat proteins organized in a database containing atomic models and information on compatible interfaces (grey area). Building blocks within pairs are described through a combination of rotation (R) and translation (T) that places a module X (defined by its constituent points p1, p2, …pN) respect to single building blocks. This information is collected in an abstraction database and employed to design modular structures according to design specifications. A genetic algorithm produces, scores and ranks designs, and the best scoring ones are converted into full atom representations. Upon energy minimization, designs are accepted or rejected based on root mean square deviation (RMSD) from the pre-minimized models.

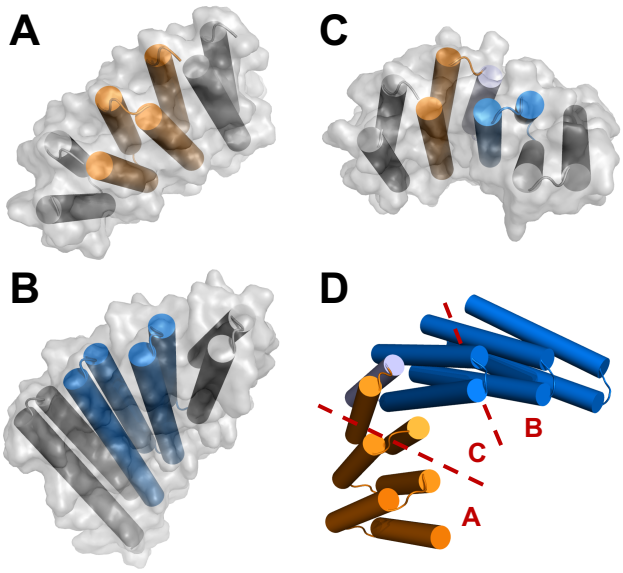


Figure 2
Repeat protein modules. A) and B) Base modules are formed by the two central repeats of designed repeat proteins. Repeats characterized by specific interfaces are depicted in orange and blue, capping repeats and protein surfaces are in grey. C) Junction modules are derived from designs were different repeat interfaces (orange and blue) are combined together through connecting elements (purple). D) Modules from A, C and B, are combined by association at conserved interfaces, indicated by the red dashed line. Repeats carrying compatible interfaces are depicted in the same color.

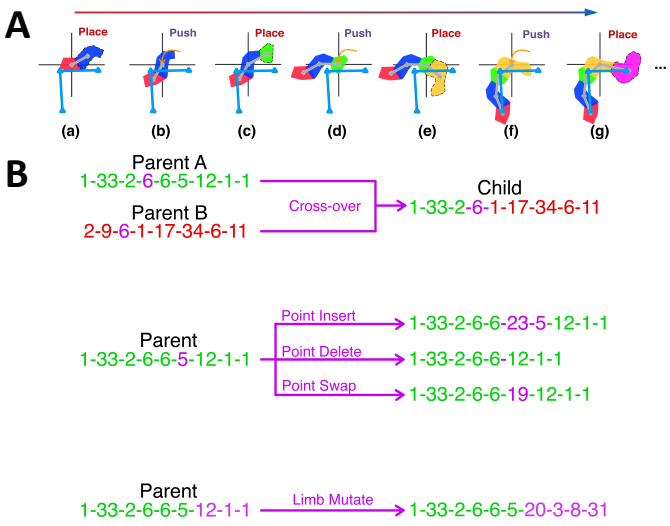


Figure 3
Abstract model building. A) Initial models are built by sequential addition of building blocks. First, a randomly selected module pair is placed with the first module on the axis origin. Then the pair is pushed so that the second module occupies the axis origin. A new random module, compatible with the last one, is placed and the whole design is pushed. The process continues until the desired length is reached. All place and push moves follow the rotation and translation information stored in the database. The CoM points used to describe the designed shape are in grey and the target shape in cyan.
B) The genetic algorithm generates diversity by crossing parent designs at a common module (cross-over), performing module point mutations (insertion, deletion or swap), if compatible with the neighbors, and removing and rebuilding whole terminal segments (limb mutate). Each number represents a specific module.

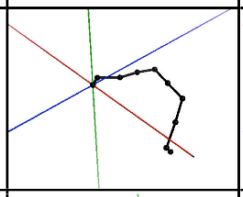
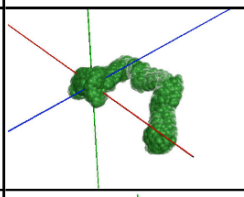
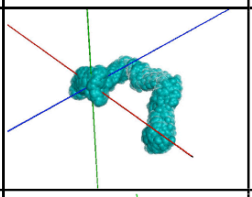
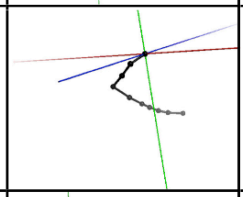
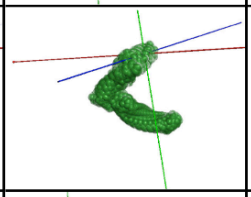
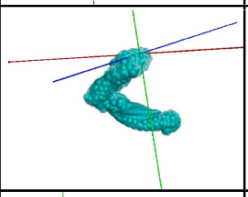
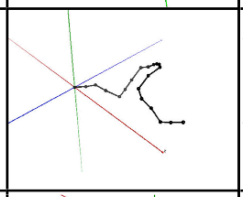
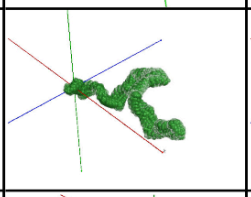
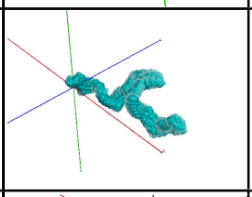
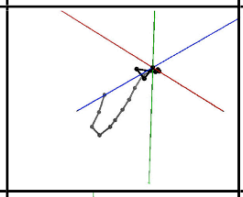
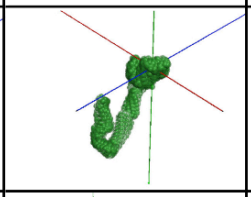
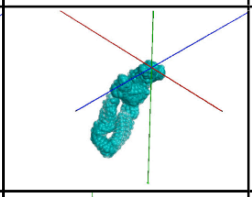
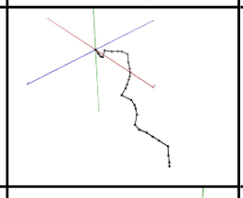
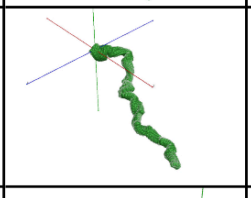
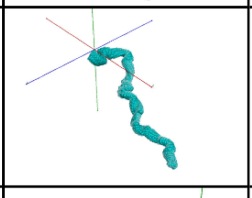
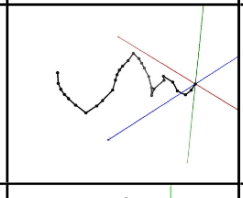
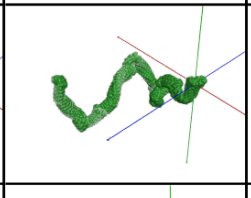
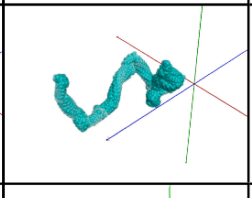
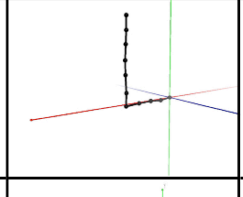
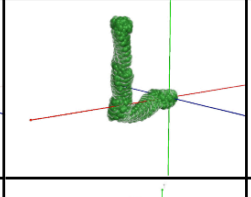
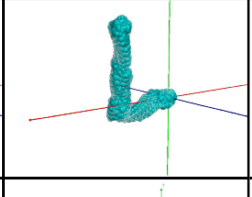
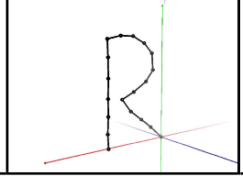
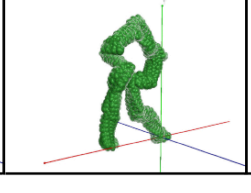
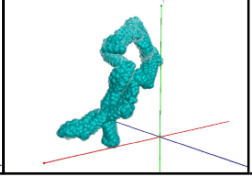
Name	Specification	Design	Relax	Max. Win. RMSD (Å)	No. of Residues
Best RL-10 (2vnd)				1.4	2082
Worst RL-10 (hfw4)				2.9	1504
Best RL-20 (1z1q)				3.9	3635
Worst RL-20 (cref)				7.0	3221
Best RL-30 (vias)				3.7	4641
Worst RL-30 (7r78)				6.6	4788
Best HD (L)				1.7	1661
Worst HD (R)				11.6	4003

Figure 4
Modular designs. Best and worst designs, according to the window RMSD score are reported for each group. RL-n is random design of length n, with n = 10, 20, 30 for the different benchmark sets. HD indicates a hand drawn shape. The design name is in parenthesis. Specification shows the target shape; design is the full atom conversion of the

GA solution; relax is the structure after Rosetta relax. Max. Win. RMSD is the highest RMSD for a 300 residues window. The last column shows the design size in number of residues.

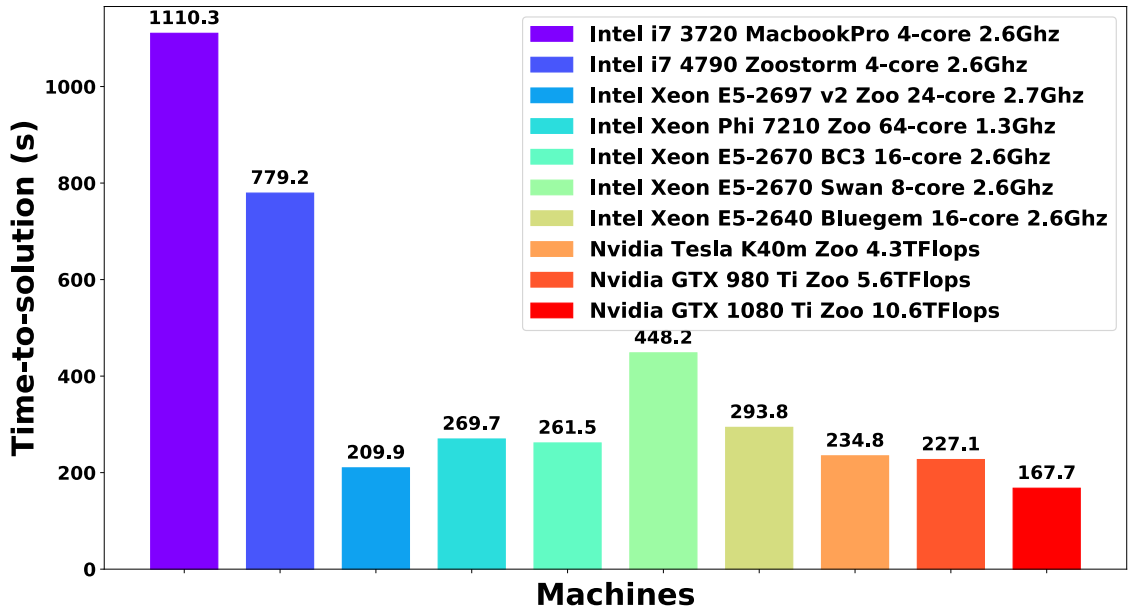


Figure 5.
Performance evaluation. Benchmark execution times in seconds for 50 iterations on the 'B' design problem. Population size was set at 524288. MacbookPro (macOS Sierra, 10.12.4) and Zoostorm (Ubuntu subsystem, Windows 10) times reflect Elfin's performance on consumer grade laptops. Ivybridge, KNL, and the Sandybridge CPUs represent a range of server CPUs. Names in brackets are computer clusters tested at the University of Bristol. Zoo runs on CentOS 7.3, Bluecrystal phase 3 (BC3) runs on Red Had Enterprise 6.3, Swan runs SUSE 12, and Bluegem runs Scientific Linux 6.6. NVIDIA K40c is a server-grade GPU compute card and the GTX 1080 is a recent high-end consumer grade dedicated GPU.